

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 10-155010

(43)Date of publication of application : 09.06.1998

(51)Int.Cl. H04L 29/08
G06F 9/46
G06F 13/00
H04L 13/08

(21)Application number : 09-263779

(71)Applicant : OKI DATA:KK

(22)Date of filing : 29.09.1997

(72)Inventor : SUDO HIDEO

(30)Priority

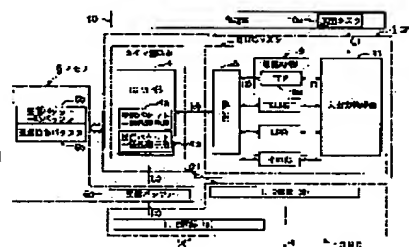
Priority number : 08258957 Priority date : 30.09.1996 Priority country : JP

(54) PACKET PROCESSING METHOD AND NETWORK ARCHITECTURE

(57)Abstract:

PROBLEM TO BE SOLVED: To reduce an overhead of an OS by processing a network layer and a transport layer with one function as a reception packet simultaneous processing means so as to process all packets received by a reception buffer through one start of protocol processing.

SOLUTION: A reception packet simultaneous processing means 4a of a TCP/IP section 4 is started at timer interrupt and processes simultaneously protocol processing relating to a network layer and a transport layer of a packet received in a reception buffer 5 within an interrupt time to store the received packet assigned in a memory 5 to a temporary buffer 5b and to post it to an IFC task 6. Furthermore, a transmission packet simultaneous processing means 4b sends a reply packet or the like stored in a transmission wait buffer 5c assigned in the memory 5 to a network within the interrupt time. The TCP /IP section 4 is operated by using a TCP and an IP as one task without conducting them as separate tasks.



LEGAL STATUS

[Date of request for examination] 29.01.2001

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

BEST AVAILABLE COPY

【特許請求の範囲】

【請求項1】 ネットワークのノード間にコネクションを確立し、受信バッファに受信したパケットに階層的に付加されたプロトコルを下位層から上位層に向かって順に処理するパケット処理方法において、

少なくともネットワーク層、トランスポート層、セッション層、プレゼンテーション層、アプリケーション層に関するプロトコルの処理をそれぞれ一つのタスクとし、ネットワーク層、トランスポート層を一つの機能とし、セッション層、プレゼンテーション層、アプリケーション層を一つの機能とし、その機能に処理の優先度を持たせて動作するようにしたことを特徴とするパケット処理方法。

【請求項2】 ネットワークのノード間にコネクションを確立し、受信バッファに受信したパケットに階層的に付加されたプロトコルを下位層から上位層に向かって順に処理するネットワークアーキテクチャにおいて、一定時間毎に入るタイマ割込みを使用し、上記受信バッファに受信されたパケットのネットワーク層とトランスポート層とに関するプロトコルを割込み中に一括処理して蓄積し、プロトコルの内容から上位層のプロトコル処理手段を指定通知する受信パケット一括処理手段と、それぞれのコネクションに属する最後の受信パケットに対する応答パケットを上記割込み中にネットワークに一括して送信する送信パケット一括処理手段とを備えたことを特徴とするネットワークアーキテクチャ。

【請求項3】 上記送信パケット一括処理手段は、上記受信パケット一括処理手段が受信パケットを一括処理中に上位層の処理手段から発生した応答パケット送信要求を、次のタイマ割込み時にまとめて送信を行い、それに対する受信側からの応答のタイムアウトをタイマ割込み回数でカウントするタイマ処理手段を備えた請求項2記載のネットワークアーキテクチャ。

【請求項4】 ネットワークのノード間にコネクションを確立し、受信バッファに受信したパケットに階層的に付加されたプロトコルを下位層から上位層に向かって順に処理するネットワークアーキテクチャにおいて、メッセージに基づいて呼び出した処理手段に起動を駆けるオペレーティングシステム機能によるメッセージ処理手段と、メッセージ処理手段により呼び出されて受信バッファから受信パケットを取り出し、データリンク層、ネットワーク層、トランスポート層に関するプロトコルを処理して所定量に達するまで蓄積する第1のパケット処理手段と、セッション層、プレゼンテーション層、アプリケーション層に関するプロトコルを処理して上位層に出力する第2のパケット処理手段と、第1のパケット処理手段から指定された第2のパケット処理手段を呼び出すメッセージをメッセージ処理手段に出力し、メッセージ処理手段からの呼び出しに応じて第2のパケット処理手段を選択するソケット処理手段とを設けたインター

フェース制御手段と、

パケット受信時に起動し、受信パケットを受信バッファに格納する毎にメッセージをインターフェース制御手段に出力する受信割り込み手段とを備えたことを特徴とするネットワークアーキテクチャ。

【請求項5】 上記第1のパケット処理手段は、上記受信割り込み手段のメッセージに基づいて呼び出され、受信バッファから受信パケットを取り出してデータリンク層のプロトコルを処理するネットワークドライバを有する請求項4記載のネットワークアーキテクチャ。

【請求項6】 ネットワークのノード間にコネクションを確立し、受信バッファに受信したパケットに階層的に付加されたプロトコルを下位層から上位層に向かって順に処理するネットワークアーキテクチャにおいて、

メッセージに基づいて呼び出した処理手段に起動を駆けるオペレーティングシステム機能によるメッセージ処理手段と、ネットワーク層、トランスポート層に関するプロトコルを処理して所定量に達するまで蓄積する第1のパケット処理手段と、セッション層、プレゼンテーション層、アプリケーション層に関するプロトコルを処理して上位層に出力する第2のパケット処理手段と、第1のパケット処理手段から指定された第2のパケット処理手段を呼び出すメッセージをメッセージ処理手段に出力し、メッセージ処理手段からの呼び出しに応じて第2のパケット処理手段を選択するソケット処理手段とを有するインターフェース制御手段と、

パケット受信時に起動し、受信パケットを受信バッファに格納する毎にメッセージをインターフェース制御手段に出力する受信割り込み手段と、

受信割り込み手段のメッセージに基づいてインターフェース制御手段のメッセージ処理手段により呼び出され、データリンク層に関するプロトコルを処理して第1のパケット処理手段を呼び出すネットワークドライバとを備えたことを特徴とするネットワークアーキテクチャ。

【請求項7】 カウンタがカウンタアップする毎にメッセージを上記インターフェース制御手段のメッセージ処理手段に出力するインターバルタイマと、インターバルタイマのメッセージに基づいて呼び出され、上記第1のパケット処理手段により蓄積されたパケット量を監視するタイマ処理手段を上記第1のパケット処理手段に設けた請求項4記載、又は請求項6記載のネットワークアーキテクチャ。

【請求項8】 上記タイマ処理手段は、上記第1のパケット処理手段がプロトコル処理中にタイマ処理を実行し、応答パケットの送信処理等のプライオリティの低い処理に負荷係数を設定し、一定の負荷に達した時点でタイマ処理を中断し、次のタイマ処理でその続きを実行することにより、1回のタイマ処理時に処理する負荷を平均化させる請求項7記載のネットワークアーキテクチャ。

【請求項9】 上記タイマ処理部は、コネクションを増減する際に参照するテーブルを記憶するテーブル記憶手段と、前回起動時の受信バッファの使用量を格納しておく第1の受信バッファ使用量記憶手段と、上記メッセージ処理手段に起動される毎に受信バッファの使用量をセンシングして第2の受信バッファ使用量記憶手段に格納するバッファ量センス部と、第1の受信バッファ使用量記憶手段の内容と第2の受信バッファ使用量記憶手段の内容とテーブル記憶手段のテーブルとを参照して増減するコネクション数を判断する判断部と、判断部の出力に基づいて許容コネクション数を設定するコネクション数設定部とを備えた請求項7記載のネットワークアーキテクチャ。

【請求項10】 上記データバッファに格納されたデータを処理するプライオリティの低い処理手段に設けられた負荷監視部と、負荷監視部からの負荷通知を記憶する負荷通知記憶手段と、負荷通知記憶手段の内容が高負荷通知ならば上記インターバルタイマに設けられたカウンタのタイマ間隔値を大きくし、否ならばカウンタのタイマ間隔値を小さくするタイマ間隔値変更手段とを備えた請求項7記載のネットワークアーキテクチャ。

【請求項11】 上記メッセージ処理手段はメッセージを格納するECB(Event Control Block)を一つ有し、ECBとソケット処理手段の上位側待ち行列とを参照し、上位側待ち行列の登録メッセージ数が所定数以上の場合には上位側待ち行列を処理する請求項4記載、又は請求項6記載のネットワークアーキテクチャ。

【請求項12】 上記第2のバケット処理手段はデータのクライアントIDを格納する共通テーブルを有し、データを出力する際にクライアントIDを格納し、上位層からデータ送信要求を受ける際には共通テーブルに基づいてクライアントIDを解析し、同一コネクションを通じてバケットを送受信する請求項4記載、又は請求項6記載のネットワークアーキテクチャ。

【請求項13】 第1のバケット処理手段が前回の受信バケットに対する応答バケットを送信してから今回の受信バケットを受信までのバケット転送時間を算出するバケット転送時間算出手段と、第2のバケット処理手段から単位時間にデータを上位層に転送する転送データ数を計数する転送データ数計数手段と、バケット転送時間と転送データ数と現在の受信バッファ空容量とから次のバケットブロック転送において受信可能なデータ容量を算出する受信可能データ容量算出手段とを設けた請求項4記載、又は請求項6記載のネットワークアーキテクチャ。

【請求項14】 上記受信割り込み手段は、上記受信バッファの「常時使用エリア」と「常時空きエリア」とに受信したバケットの情報を格納する共通テーブルを有し、バケットを受信する毎に共通テーブルを参照して「常時空きエリア」に受信したバケットが送信側から受

信側に送信される受信可能確認用ポーリングバケットの場合には応答バケットを作成して送信し、受信可能確認用ポーリングバケット以外の場合にはそのバケットを破棄する請求項4記載、又は請求項6記載のネットワークアーキテクチャ。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】 本発明はネットワークのノード間にコネクションを確立し、受信バッファに受信したバケットに階層的に付加されたプロトコルを下位層から上位層に向かって順に処理するバケット処理方法とネットワークアーキテクチャとに関する。

【0002】

【従来の技術】 従来、ネットワークのノードに組込まれるOS I(Open System Interconnection) 環境の標準ネットワークアーキテクチャは、プロトコルを階層的に並べた構造を採っており、ISO(International Organization for Standardization)等により、各層の位置付けや、機能、層間のインタフェースなどが細かく規定されている。

【0003】 ISOが規定したOS I参照モデルには、下位から物理層、データリンク層、ネットワーク層、トランスポート層、セッション層、プレゼンテーション層、アプリケーション層の7層構造になっている。

【0004】 ネットワークに送り出されるバケット形式のデータには、それぞれ、各層に関する制御情報であるプロトコルがヘッダとして付加されており、各ノードのネットワークアーキテクチャはネットワークからバケットを受信してバケットのヘッダに階層的に含まれるプロトコルを各層で順次処理してバケットに含まれるデータをデータバッファに格納するとともに、受信バケットに対する応答バケットをネットワークに送信する。

【0005】 例えば、インターネットプロトコル層(以後IPと記す)はネットワーク層に位置し、主に一つ又は複数の通信網を介して中継を行い、始点となるノードから終点となるノードまでのデータ通信を行う。

【0006】 また、IPの上位層(トランスポート層)としてのトランスファコントロールプロトコル層(以後TCPと記す)は、経路選択や中継機能に関与せず、始点及び終点となるノードのさらに終端間における透過的な両方向同時転送機能を提供する。

【0007】 TCPをネットワークアーキテクチャに組込むには、一つのコネクション(データを転送するための論理的な通信路)に対する処理を一つのTCPタスクで行せるようにし、N個のコネクションに対して(N+1)個のTCPタスクを起動させる。TCPタスクの残りの一つはマスタとして次のコネクションの確立用として待機している。

【0008】 IPは、1つのタスクが、受信した受信バケットのヘッダ処理を順次処理して対応するTCPタス

クを識別し、待機しているTCPタスクに預けていく処理を行っている。よって確立するコネクション数とは関係なく常に1つのタスクとして存在する。

【0009】各TCPタスクはそれぞれコネクションを確立したクライアントタスクから受信した受信パケットを処理し、相手のクライアントタスクに応答パケットを送信する。このとき送信する応答パケットはただ一つ存在するIPタスクを通る必要があるため、各TCPタスクはIPタスクの使用権を獲得する。

【0010】IPタスクはTCPタスクまたはデータリンク層タスクからの呼び出し（以後POSTと記す）を待つ。POSTとは、マルチタスク環境下のタスク間で処理を移行する時に使用するオペレーションシステム（以後OSと記す）のシステムコールである。

【0011】ここで、簡単にIPタスクとTCPタスクとの動作を説明する。データリンク層タスクは、クライアントより最初のパケットを受信すると、そのヘッダからIPタスクを識別し、待機しているIPタスクに対してPOSTを行う。

【0012】IPタスクは受信パケットのヘッダからネットワーク層に関するプロトコル処理を行い、ヘッダからTCPタスクを識別し、待機しているTCPマスタタスクに対してPOSTを行う。

【0013】TCPマスタタスクは受信パケットの内容がコネクションの開設要求と判断すると、パケットを処理し、クライアントに対して応答パケットを送信する。通常クライアントからの最初のパケットはコネクションの開設を要求するパケットなので、クライアントへ応答パケットを送信した後はタイマを動作させてスリーウェイハンドシェークの最後のパケットを待つ。

【0014】TCPマスタタスクは、タイマアップする前にクライアントからハンドシェークパケットを受信すると、コネクションの確立を認識する。また、コネクション確立後の受信パケットがセッション層以上のサービスの要求がある場合には、新たにマスタタスクを生成しPOSTを行う。本例では、FTP（File Transfer Protocol）マスタタスクを起動する。

【0015】FTPマスタタスクは、クライアントとの間に制御コネクションを確立すると、fork()関数を用い自分自身のコピー（FTPタスク）を作成する。FTPタスクは、TCPマスタタスクに対してポートの割り当てを行う。

【0016】TCPマスタタスクは、fork()関数を用い自分自身のコピー（TCPタスク）を作成する。TCPタスクは、FTPタスクとの間にデータコネクションを確立し、以後クライアントから切断されるまでそのコネクションを管理する。

【0017】TCPマスタタスクは次のクライアントからのコネクションの開設要求に備えて待ち状態となる。

【0018】他方、TCPタスクはパケットの受信毎に

トランスポート層の処理を行い、FTPタスクに対してPOSTを行う。FTPタスクは、TCPタスクから受信したデータを受け取り、データ中に含まれるクライアントからのコマンドを実行する。

【0019】以上説明した組込み処理は、完全にプリエンブティブに動作するラウンドロビン方式のOSの組込みを前提としている。

【0020】

【発明が解決しようとする課題】従来のネットワークアーキテクチャにあっては、複数のコネクションをそれぞれ同数のTCPタスクで処理するため、コネクションを処理するプロセスを動的に複数コピーする機能（上述したfork()関数）が必要となる。この機能は1つのプログラムを複数のタスクで共有し実行させるため、タスク間の同期を取るラウンドロビン方式の完全にプリエンブティブに動作する高機能なOSを必要とするという問題点があった。

【0021】また、各層のプロトコル処理を別々のタスクとして動作させるとともに、トランスポート層以上のタスクはコネクションの確立に応じてタスクがコピーされるので、OS上で動作するタスクの数が増え、OSのオーバーヘッド量が増加するという問題点もあった。

【0022】また、プロトコル処理における各プロセスをすべて別々のタスクが行うことにより、メモリ上の作業領域が増大するという問題点もあった。

【0023】本発明はネットワークアーキテクチャに組み込まれるネットワーク層／トランスポート層の負荷を軽減し、より効率良く動作するパケット処理方法とネットワークアーキテクチャとを提供することを目的とする。

【0024】

【課題を解決するための手段】上記目的を達成するために本発明のネットワークアーキテクチャにおいては、一定時間毎に入るタイマ割込みを使用し、受信バッファに受信されたパケットのネットワーク層とトランスポート層とに関するプロトコルを割込み中に一括処理して蓄積し、プロトコルの内容から上位層のプロトコル処理手段を指定通知する受信パケット一括処理手段と、それぞれのコネクションに属する最後の受信パケットに対する応答パケットを割込み中にネットワークに一括して送信する送信パケット一括処理手段とを備える。

【0025】また、同じ目的を達成するために本発明のパケット処理方法においては、少なくともネットワーク層、トランスポート層、セッション層、プレゼンテーション層、アプリケーション層に関するプロトコルの処理をそれぞれ一つのタスクにし、ネットワーク層、トランスポート層を一つの機能とし、セッション層、プレゼンテーション層、アプリケーション層を一つの機能とし、その機能に処理の優先度を持たせて動作するパケット処理方法とし、具体的手段として、メッセージに基づいて

呼び出した処理手段に起動を駆けるオペレーティングシステム機能によるメッセージ処理手段と、メッセージ処理手段により呼び出されて受信バッファから受信パケットを取り出し、少なくともネットワーク層、トランスポート層に関するプロトコルを処理して所定量に達するまで蓄積する第1のパケット処理手段と、セッション層、プレゼンテーション層、アプリケーション層に関するプロトコルを処理して上位層に出力する第2のパケット処理手段と、第1のパケット処理手段から指定された第2のパケット処理手段を呼び出すメッセージをメッセージ処理手段に出力するとともに上位側待ち行列に登録し、メッセージ処理手段からの呼び出しに応じて上位側待ち行列を参照し、第2のパケット処理手段を呼び出すソケット処理手段とを備える。

【0026】

【発明の実施の形態】本発明の実施の形態について図面を参照しながら説明する。尚、各図面に共通な要素には同一符号を付す。

第1の実施の形態

図1は第1の実施の形態によるプリンタのネットワークアーキテクチャを示すブロック図であり、単一の中央処理装置1（以後CPU1と記す）上で動作する割り込み処理及びタスクを示す。

【0027】尚、ノードとして、図2に示すように、対象を回線に接続されたネットワークプリンタ2（以後プリンタ2と記す）にしほり説明する。

【0028】ネットワークアーキテクチャは、ネットワーク接続用ハードウェアとしてのNIC（Network Interface Controller）3と、トランスポート層及びネットワーク層のプロトコル処理手段としてのTCP/IP部4と、クライアントとのデータの送受信を制御し、メモリ5上に割り当てられた受信バッファ5aを管理するIFCタスク6とからなる。

【0029】NIC3はネットワークからのパケットの受信時に起動し、CPU1に割り込みをかけて受信パケットを受信バッファ5に受信し、受信パケットに対する応答パケット等を送信する時に使用される。

【0030】TCP/IP部4はタイマ割り込み時に起動し、割り込み時間内に受信バッファ5に受信されてある受信パケットのネットワーク層とトランスポート層とに関するプロトコル処理を一括処理してメモリ5上に割り当てられた受信パケット一時バッファ5bに格納してIFC（Interface Control）タスク6にPOSTを行う受信パケット一括処理手段4aと、メモリ5上に割り当てられた送信待ちバッファ5cに格納されてある応答パケット等を割り込み時間内にネットワークに送信する送信パケット一括処理手段4bとが設けてある。TCP/IP部4は、TCPとIPとを別タスクとせず一つのタスクとして動作する。

【0031】IFCタスク6は、TCP/IP部4によ

り指定された通信アプリケーションを識別する通信アプリケーション識別部8と、受信パケットのセッション層、プレゼンテーション層、アプリケーション層に関するプロトコルを処理する通信アプリケーション群9と、受信パケットに含まれる受信データを通信アプリケーション群9から入力して印刷部10に出力する入出力管理部11とからなる。

【0032】図3は図1に示したネットワークアーキテクチャの処理時のタイムチャートである。TCP/IP部4は時間も経過する毎にタイマ割り込みに入るように設定されており、割り込み時間内にTCP/IP部4のプロトコル処理を行う。時間もは割り込み処理時間に比べて大きく設定されており、TCP/IP部4が割り込み処理を終了するとPOSTを行い、以後、次のタイマ割り込みに入るまでNIC3によるパケット送受信処理、IFCタスク6によるパケット処理、印刷部10による印刷処理等が行われる。

【0033】TCP/IP部4はタイマ割り込みにより起動すると、受信バッファ5に受信済みの全パケットを割り込み時間内に処理し、最後のパケットに対する応答パケットを作成して送信待ちバッファ7bに格納する。たとえば、受信バッファ5にクライアントAから3つのパケットA1～A3を受信してあるとすると、1回のタイマ割り込み処理で3つのパケットA1～A3を処理し、パケットA3に対する応答パケットを作成して送信待ちバッファ7bに格納する。

【0034】また、複数のクライアントからパケットを受信した場合にも同様に、各クライアントから受信した最後のパケットに対して応答パケットを作成して送信待ちバッファ7bに格納する。つまり、クライアントAから3つのパケットA1～A3を、クライアントBから1つのパケットB1を、クライアントCから2つのパケットC1、C2を受信バッファ5に受信していた場合、TCP/IP部4は、クライアントAにはパケットA3に対する応答パケットを、クライアントBにはパケットB1に対する応答パケットを、クライアントCにはパケットC2に対する応答パケットをそれぞれ作成して送信待ちバッファ7bに格納する。

【0035】送信処理はNIC3に対して送信に必要な情報をセットする前処理、NIC3が行う送信処理、送信完了後行う後処理からなる。

【0036】送信パターンの1つ目は、TCP/IP部4から応答される場合で、TCP/IP部4とのコネクション確立シーケンス時の応答、データパケットに対する応答、ARP（Address Resolution Protocol）における物理アドレスの問い合わせに対する応答等、受信パケットを受信した直後に発生する送信処理であり、上述したタイマ割り込み処理中に行う。

【0037】即ち、データパケットのストリームに対する単なる応答である場合には、上述した通りに直ちに送

信を行わず、受信済みの全パケットの処理が終了した後に応答パケットを作成する。その他の応答の場合には、直ちに送信用のパケットを作成して送信待ちバッファ7bに格納し、タイマ割り込み処理の最後にまとめて送信要求をNIC3に出す。

【0038】こうすることにより、送信パケットに対する応答を受信するまでのタイムアウト時間(ターンアラウンド時間)を一括に管理する(以後Max Time out処理と記す)ことができ、タイムアウト処理にかかる負荷が減少する。

【0039】本実施の形態によるアーキテクチャ組込みにおけるタイマ処理は、各コネクションを管理する情報とともにタイマ情報をグローバルテーブルとして管理する。送信パケットのタイムアウト値、タイムアウトカウンタ等をテーブル中に保存し、タイマ割り込み中でカウンタをデクリメントし、タイマ割り込みの入回数でタイムアウト時間を管理する。タイムアウト値はTCP/IP部4のタイマ割り込みが入る時間tの倍数で定義し、tの倍数単位の増減を行う。

【0040】送信パターンの2つ目として、印刷部10のプリンタファームウェアや通信アプリケーション群9の通信アプリケーション等から任意の時間に発生する送信処理がある。

【0041】従来のアーキテクチャ組込みによる送信では、TCPタスク及びIPタスクは、IFCタスク6から送信要求を受けると、図4(A)に示すように、直ちに送信処理を行い、NIC3へ送信処理を要求した。

【0042】しかし、アプリケーションの送信では、タイムアウトが存在しないか、存在しても非常に長いので緊急に送信する必要はない。従って、本発明ではIFCタスク6の送信要求を一定時間蓄積し、一括して送信する方法を採る。

【0043】通信アプリケーション関数内及びプリンタファームウェア内で送信要求が発生すると、通信アプリケーション関数が、特定のテーブルに対して送信要求を書き込む。実際の送信は、図4(B)に示すように、TCP/IP部4が次に入るタイマ割り込み中に通信アプリケーションによってセットされたテーブル情報を参照して行う。

【0044】また、タイマ割り込み時に受信パケットが受信バッファ5内に存在する時は、受信パケットの処理を優先して行い、その応答パケットと一緒に送信要求をNIC3に対して行う。

【0045】次に動作について説明する。クライアントよりランダムに送られてくるパケットは、図1に矢印(a)、(b)で示すように、第1、2層部のNIC3によりリアルタイムに受信される。NIC3はパケットを受信する毎にCPU1に割り込みをかけて、割り込み処理によりパケットを受信バッファ5に格納する。

【0046】TCP/IP部4は、時間t毎に入るタイ

マ割り込み中にネットワーク層及びトランスポート層のプロトコル処理を、図1に矢印(c)で示すように、その時受信バッファ5に受信してある全パケットについて行い、処理済みのパケットをパケット一時バッファ7aに格納する。

【0047】一連の受信データストリームにおける最終パケットを受信すると、TCP/IP部4は、ヘッダから上位層プロトコル(通信アプリケーション)を識別し、それを通知するフラグをセットして、図1に矢印(d)で示すように、IFCタスク6に対してPOSTを行う。

【0048】フラグはグローバルな変数で、TCP/IP部4が受信データストリームを渡すべき上位層(通信アプリケーション)を指定するために用い、TCP/IP部4からIFCタスク6に対して受け渡される。受け渡し方法は、OSのメッセージパッシング機能を用いる。

【0049】通信アプリケーションは、すべてIFCタスク6の関数として組み込まれ、指定された上位層が、例えば、FTP9aとすると、図1に矢印(e)で示すように、識別部8はFTP関数をコールする。

【0050】FTP9aは、TCP/IP部4が受信したデータストリームを受信パケット一時バッファ7aから受け取り、内容を処理する。必要であれば、図1に矢印(f)で示すように、入出力管理部11へデータを渡す。入出力管理部11へ渡すデータは、コネクション別になっているので、コネクション別のキュー(以後、IFC-EDIT間受信キューと記す)を作成する。

【0051】IFC-EDIT間受信キューは、IFCタスク6の通信アプリケーションFTP9aから入出力管理部11を介し、印刷部10のEDIT(編集)タスク10aへ印刷データ、その他を渡す待ち行列である。印刷部10のEDITタスク10aは、図1に矢印

(g)で示すように、IFC-EDIT間受信キューからデータを受け取り編集処理を行う。

【0052】ここで、従来の方法(Aとする)による受信処理時間rAと、本実施の形態方法(Bとする)による受信処理時間rBとの差Recを以下に示す。

【0053】まず、処理をTCP/IP部4に、時間をt内に限定し、パケット受信時間を、パケット処理時間とOSの処理時間に分けて求める。単位時間に受信するパケットの数をn、1パケットを処理する時間をEとすると一定時間tにおけるパケットの処理時間E(A, B)は、A, B共に、

$$E(A, B) = E \times t \times n \cdots \cdots (1)$$

また、OSの処理時間は、タスク生成時間とタスクスイッチ時間との和とすると、fork()関数等により子タスクを作成し実行させる時間をF、同時にアクセスする平均コネクション数をNとすると一定時間tにおけるタスク生成処理(コネクション確立毎生成)時間F(A)は、

Aのみ該当し、

$$F(A) = F \times N \dots \dots \dots (2)$$

タスクスイッチに要する時間 (POSTされてからタスクが起動するまでの時間) において、従来の組込みにおいて対象となるOSをPA、割込みによるマルチタスクOSをPBとすると1パケット受信に対しIPタスク、TCPタスクがそれぞれ起動するため一定時間tにおけるタスクチェンジ処理時間P(A)、P(B)は、Aでは、1パケット受信毎にIPタスク、TCPタスクが呼び出されるので、

$$P(A) = 2 \times PA \times t \times n \dots \dots \dots (3)$$

Bでは、t時間内に1度TCP/IP部4が呼ばれるので、

$$P(B) = PB \dots \dots \dots (4)$$

従来方法の一定時間tにおける受信処理に要する時間rAは式(1)(2)(3)より、

$$rA = E(A, B) + F(A) + P(A) \\ = Et_n + FN + 2(PA)t_n \dots \dots (5)$$

タイマ割り込みが起動し、割り込み内の処理が起動するまでの時間をIとすると、本方法の一定時間tにおける受信処理に要する時間rBは、(1)(4)より、

$$rB = E(A, B) + P(B) + I$$

$$SA = t \times n \times R + t \times n \times PA + t \times n \times S \dots \dots (8)$$

$$SB = N \times R + S \dots \dots \dots (9)$$

式(8)(9)より、

$$Res = SA - SB$$

$$= t_n R + t_n (PA) - NR + (t_n - 1) S \dots \dots (10)$$

nは単位時間に受信するパケット数、tは本実施の形態による組込みで採用するタイマ割込みの間隔、PAは従来の組込みにおいて対象となるOSのタスクスイッチ時間、PBは割込みによるマルチタスクOSのタスクスイッチ時間、Nは同時にアクセスする平均コネクション数、Rは1パケットの応答処理に要する時間、SはNIC3への送信要求処理時間と送信完了割り込み処理時間の和である。従って、本実施の形態による組込み処理によって応答処理時間を $t_n R + t_n (PA) - NR + (t_n - 1) S$ だけ短縮することができる。

【0058】また、図7に示すように、実際の送信では、NIC3に対して送信に必要な情報をセットする前

$$YA = t \times Y \times S \dots \dots \dots (11)$$

$$YB = S \dots \dots \dots (12)$$

式(11)(12)より

$$Sen = YA - YB$$

$$= (tY - 1) S \dots \dots \dots (13)$$

Yは単位時間内に発生する送信要求数、SはNIC3への送信要求処理時間と送信完了割り込み処理時間の和である。

【0060】一定時間t内にパケットを受信している場合は、その応答パケットと一緒に送信処理を行うため、YBは0となり、 $Sen = tYS$ となる。従って、本実

$$= Et_n + PB + I \dots \dots \dots (6)$$

となるから、それらの差Recは式(5)、(6)より次の式で示すことができる。

【0054】

$$Rec = rA - rB$$

$$= FN + 2(PA)t_n - PB - I \dots \dots (7)$$

従って、本組込み方法により一定時間tにおけるパケットの受信時間を $FN + 2(PA)t_n - PB - I$ だけ短縮することができる。

【0055】また、従来の組込み方法では、受信側ノードは1パケットを受信するとIPタスクを経由し、直ちに制御をTCPタスクへ移す。そしてTCPタスクが1パケットの処理を終了すると、制御を他のタスクへ移さなければならないので、後続のパケットの受信を待って、それらをまとめて応答することは難しい。そのため、1パケット受信する毎に応答パケットをクライアントに返していた。

【0056】従来の方法の応答処理時間SAと本実施の形態による方法の応答処理時間SBとの差Resは次のように求めることができる。

【0057】

処理、NIC3が行う送信処理と、送信完了後行う後処理が必要である。特に後処理は、送信完了時に割込みとして起動されるのでレジスタの退避等で数100ミリ秒が必要となり、従来の方法では送信の度にこの割込みが発生する。本実施の形態による方法では時間t内に発生する送信要求および、受信したデータの応答パケットの送信を、NIC3に対し同時に起動をかけるため送信における前処理と後処理は常に1回となる。

【0059】送信パターン2の一定時間tにおける、従来の組込み方法の処理時間YA、と本実施の形態による組込み方法の処理時間YBとの差Senを求めると、

施の形態による組込み方法により、送信時間を $(tY - 1)S$ または、 tYS だけ短縮することができる。

【0061】本送信処理で送信するパケットのタイムアウト処理も送信パターン1と同様にタイマ割り込み中で行う。従来の送信したパケットに対するターンアラウンド時間のタイムアウト処理は、送信したすべてのパケッ

トそれぞれに対しタイマをカウントする必要があるが、本実施の形態による方法のように一定数のパケットを同時に送信することにより、パケットのタイマ管理をグループ化し処理時間を短縮することができる。

【0062】また、従来はコネクションの接続している間はTCPタスクは起動したままとなり、タイムスライス処理で他のタスクを実行しており、タイムスライス処理に常に一定量のCPU資源を使用しているため、プロトコル処理や、プリンタファームウェアの処理の動作する効率が非常に悪くなる。

【0063】本実施の形態による組込み方法では、ジョブのスケジューリングにプライオリティを用いた割り込みを使用し、タイムスライスによるラウンドロビン方式は採用していない。

【0064】第1の実施の形態によれば、TCP/IP部は受信バッファに受信してあるすべてのパケットを割り込み処理時間内に処理するので、OSのオーバーヘッドを削減することができ、CPUを他の有効な処理に割り当てることができる。

【0065】第2の実施の形態

一般に割り込み内での動作が多いと、他の処理への影響が大きくなるので、第2の実施の形態では極力割り込み内の処理を減らし、大部分の処理をIFCタスクに組み込むようにしたものである。

【0066】図5は第2の実施の形態によるプリンタのネットワークアーキテクチャを示すブロック図であり、受信割込み手段20とインターバルタイマ21とインターフェース制御手段としてのIFCタスク22とから成る。

【0067】受信割込み手段20は、パケットの受信時に起動して受信パケットをメモリ5の受信バッファ5aに格納する毎にIFCタスク22にメッセージ(message-X)を出力する。インターバルタイマ21はカウンタ21bがカウントアップする毎にIFCタスク22にメッセージ(message-S)を出力する。

【0068】IFCタスク22は、メッセージを受信して呼び出し処理を行うメッセージ処理手段24と、データリンク層(2層)、ネットワーク層(3層)、トランスポート層(4層)の各プロトコルを処理して所定量に達するまでメモリ5のパケット一時バッファ5bに蓄積する第1のパケット処理手段23と、パケット一時バッファ5bからパケットを取り出してセッション層(5層)、プレゼンテーション層(6層)、アプリケーション(7層)の各プロトコルを処理し、データをメモリ5のデータバッファ5dに出力する第2のパケット処理手段30と、第1のパケット処理手段23から呼び出されてメッセージ(message-A)をメッセージ処理手段24に出力し、メッセージ処理手段24からの呼び出しに応じて第2のパケット処理手段30に起動を駆けるソケット処理手段28と、セントロニクス、RS-232C等のケ

ーブルを制御するIFCタスク内の制御モジュール32とを備えている。

【0069】第1のパケット処理手段23は、受信割込み手段20からメッセージ処理手段24にメッセージ(message-X)が出力されることによって起動され、受信バッファ5aから受信パケットを取り出してデータリンク層(2層)のプロトコル処理を行うネットワークドライバ25と、ネットワークドライバ25が処理した受信パケットのネットワーク層(3層)とトランスポート層(4層)の各プロトコルを処理してメモリ5のパケット一時バッファ5bに蓄積するプロトコルスタック(TCP/IP、SPX/IPX、その他)26と、インターバルタイマ21からメッセージ処理手段24にメッセージ(message-S)が出力されることによって起動され、コネクションを管理するタイマ処理手段31とを備えている。

【0070】第2のパケット処理手段30は、プロトコルスタック26を使用することにより実際に他のノードと通信するアプリケーション(FTP、RPRINTER、その他)27と、アプリケーション27が処理したデータのキュー管理を行うジョブ制御モジュール29とを備えている。

【0071】ソケット処理手段28はアプリケーション27とプロトコルスタック26の間の通信を行う上で共通のインタフェースを提供し、プロトコルスタック26に呼び出され、アプリケーション27を指定するメッセージ(message-A)をメッセージ処理手段24に出力するソケット(Low)28aと、メッセージ処理手段24に呼び出されメッセージに含まれる内容に基づいてアプリケーション27を選択するソケット(High)28bとを有する。

【0072】図6はアプリケーションFTPの構成を示すブロック図である。アプリケーションFTP27aは、イニシャル処理部35と、メイン処理部36とから構成される。

【0073】イニシャル処理部35は、プリンタ1からのイニシャル要求に対し、FTP内のワーキング等の初期化の他、アプリケーション登録をソケット(High)28bに対して行う。

【0074】メイン処理36は、メッセージ(message-A)の解析部37と、下位層の状態をmessage-Aによって通知された時に呼び出されるコネクション管理部38と、受信したパケットの処理を行う際に参照するデータ解析部39と、データ解析部39でコマンドと判断されたときに呼ばれるコマンド分岐部40と、各コマンドの処理群41とから成る。

【0075】図7はコネクション管理テーブルの説明図であり、コネクション別に「コネクションの状態」、「処理中のコマンド」、「コネクションNo」、「送信パケット数」、「相手先ポートNo」が配列されている。コネクション管理テーブルはコネクション管理部3

8により管理され、受信したパケットの処理を行う際にメッセージ解析部37より参照される。

【0076】図8はコネクションの状態と設定値との関係を示す説明図、図9はプロトコルスタックの各処理手段の処理ステップ数と負荷との関係を示す負荷算出テーブル、図10はパケットを再送する際の処理を示すフローチャートである。

【0077】次に動作について説明する。クライアントよりランダムに送られてくるパケットは、ネットワーク接続用H/WであるNIC3からの受信割り込み信号がCPU1に inputs され、受信割り込みプログラムからなる受信割り込み手段20によってリアルタイムに受信され、受信バッファ5aに格納される。1パケットを格納する毎に受信割り込み手段20は、図5に矢印(a)で示すように、メッセージ(message-X)をIFCタスク22に対し出力する。

【0078】IFCタスク22のメッセージ処理手段24は、アイドル時はIFCタスク宛のメッセージのセンシングをしている。メッセージ(message-X)よりも先にIFCタスク宛に送られたメッセージがあればIFCタスク22はその処理を先に実行する。

【0079】メッセージ処理手段24はメッセージ(message-X)を受信すると、図5に矢印(b)で示すように、ネットワークドライバ25を呼び出す。ネットワークドライバ25は、通常のデータリンク層(2層)の処理を行うが、複数のNIC3および複数の2層プロトコル(例えばEthernet、IEEE802.3、IEEE802.5等)にも対応できる。

【0080】ネットワークドライバ25は、データリンク層のプロトコル処理を終了すると、図5に矢印(c)で示すように、上位層を識別して対応するプロトコルスタック26、例えばTCP/IP26aを呼び出す。TCP/IP26aはTCPとIPとを別タスクとせず一つのタスクとして作成されているので、TCP-IP間のオーバーヘッドを削減する。

【0081】TCP/IP26aは、ネットワークドライバ25から受信したパケットのネットワーク層(3層)とトランスポート層(4層)のプロトコル処理を行い、そのパケットをメモリ5のパケット一時バッファ5bに格納し、ネットワークドライバ25に対して制御を返す。

【0082】ネットワークドライバ25は制御をIFCタスク22から他のタスクへ返し、次のパケットの受信を待つ。

【0083】以上の処理を数回繰り返すと、パケット一時バッファ5bにパケットが一定量蓄積される。TCP/IP26aはパケットが一定量に達すると、図5に矢印(d)で示すように、ソケット(Low)28aに対し、対応するアプリケーション27を指定する情報を出力してパケットの受け取りを要求する。

【0084】ソケット(Low)28aは、直接、ソケット(High)28bを呼び出さず、図5に矢印(e)で示すように、メッセージ処理手段24に対しメッセージ(message-A)を出力する。本処理により、メッセージ処理手段24に起動を駆けられ、セントロニクス、RS-232C等のケーブルを制御するIFCタスク内の制御モジュール32が動作するきっかけを与えることができる。

【0085】メッセージ処理手段24はメッセージ(message-A)を受信すると、図5に矢印(f)で示すように、ソケット(High)28bを呼び出す。ソケット(High)28bは、図5に矢印(h)で示すように、メッセージ(message-A)の内容に基づいてアプリケーション、例えばFTP27aを呼び出す。

【0086】FTP27aは、パケット一時バッファ5bに蓄積されたパケットを全て処理し、図5に矢印(i)で示すように、ジョブ制御モジュール29を呼び出す。

【0087】ジョブ制御モジュール29は、図5に矢印(j)で示すように、パケットのデータをメモリ5のデータバッファ5dに格納する。

【0088】ここで、FTPの動作を図6～図11を参照して詳細に説明する。FTP27aはソケット(High)28bから受信したメッセージ(message-A)により呼ばれる関数により起動される。メッセージ解析部37はメッセージ(message-A)の内容を解析し、コネクションに関する状態を通知するものであれば、コネクション管理部38を呼び出す。コネクション管理部38は、メッセージ(message-A)の内容により、図7、8に示すコネクション管理テーブルの「コネクションの状態」の設定値を変更する。

【0089】メッセージ解析部37は、メッセージ(message-A)の内容がパケットの受信であれば、データ解析部39を呼び出す。データ解析部39は、ソケット(High)28bに対してパケットの獲得を要求し、受信したパケットの「コネクションの状態」を知るために、図7に示したコネクション管理テーブルの「コネクションの状態」を参照する。

【0090】「コネクションの状態」が、図8に示したように設定値4であれば、獲得したパケットは、印字データと判断し、ジョブ制御モジュール29にパケットを渡す。「コネクションの状態」が、設定値0、又は3である場合には、パケットを受信する状態ではないので、獲得したパケットを捨てる。

【0091】「コネクションの状態」が、設定値1、又は2である場合には、獲得したパケットは、コマンドであると判断してコマンド分岐部40を呼び出す。コマンド分岐部40は、「コネクションの状態」が、設定値1で受信したパケットがログイン要求(USERコマンド)である場合には、ログイン処理と判断し、コネクション管理部38を呼び出す。

【0092】「コネクションの状態」が、設定値2である場合には、ログイン済であるので、ログイン要求以外のコマンドを受け付けられるので、該当するコマンド処理部41を呼び出す。

【0093】コマンド処理部41は送受信を必要とするコマンドを受け付けた場合には、クライアントに対して処理し、データのコネクションの確立を行うために、下位層に対してコネクションの確立要求を出す。

【0094】コネクションが確立されるまでの間は、他の処理に制御を渡すため、データコネクションの確立後の処理をコネクション管理テーブルの「処理中のコマンド」に書き込み、保持する。

【0095】このように、各処理が独立してコネクション管理テーブルの「コネクションの状態」エリアの設定値によって動作するので、複数のコネクション管理を行うことができる。

【0096】パケット送信時のタイマ処理は、図5に矢印(1)～(n)で示すように行なわれる。まず、一定時間毎にインターバルタイマ21は、矢印(1)で示すように、IFCタスク22のメッセージ処理手段24に対してメッセージ(message-S)を出力する。

【0097】メッセージ処理手段24はメッセージ(message-S)を受信すると、矢印(m)に示すように、タイマ処理手段31を呼び出す。タイマ処理手段31は、矢印(n)に示すように、必要に応じてプロトコルスタック26を呼び出す。

【0098】タイマ処理手段31及びタイマ処理手段31によって呼び出されるプロトコルスタックの動作を図10に基づいて説明する。プロトコルスタック26は、例えばTCP/IP26aが呼び出されているものとする。まず、ステップS1とステップS2とは、全コネクションにおいて毎回実行しなければならない処理で、タイマ処理手段31を起動することにより必ず実行される。

【0099】即ち、「コネクションの状態」のチェック、応答待ちパケットのタイマカウント処理、パケットの再送処理、Max Timeout処理等を実行する。

【0100】その他のステップは、プロトコル上必ず毎回実行する必要のない処理で、処理時間のかかるものが多く、時間的に処理量のばらつきが大きい。これらの処理は、図9に示すような処理ステップ数と対応する負荷係数Gを用いる。

【0101】まずステップS3で、負荷変数を初期化する。ステップS4で、コネクション単位に要求される処理を行い、1コネクションの処理が終了するまで行う。

【0102】ステップS5で1コネクションの処理の最後に行った処理の負荷係数Gの合計を負荷変数に加算する。ステップS6で、負荷変数が一定値(MAX)以上であれば、ステップS8で処理を中断してタイマ関数を抜

る。このとき次にタイマ関数が呼び出された時に処理を開始するコネクションの位置(コネクションNo.)をメモリ5のグローバルエリア内の変数に記憶しておく。

【0103】負荷変数が一定値(MAX)以下であれば、全コネクションの処理を終了するまでステップS3～ステップS7を繰り返す。

【0104】負荷変数が一定値(MAX)以下のまま、すべてのコネクションの処理を終了した場合はタイマ処理を抜ける。

【0105】第2の実施の形態によれば、ネットワークプロトコル(2層～7層)処理をシングルタスク上でソケットインタフェースにより下位層と上位層とに分け、OSの基本機能であるメッセージによって各層の処理手段に起動をかけ得るようにしたので、シングルタスク上で2つの処理に優先順位をつけることができ、また、ネットワークプロトコル処理中に割り込みにより他のインタフェース処理を受け付けることができるので、同一タスク上で他のホストインタフェース(セントロニクス、RS-232C)との共存を可能とした。

【0106】また、タイマ処理手段を設けたことにより、一時期に集中しがちな処理の負荷を分散し平均化させ、安定したシステム状態を保つことができる。このことにより、プリンタではオーバーランや、オーバーフロー等の性能を向上させ、ネットワーク処理の集中に関係なく安定した性能を保つことができる。

【0107】また、シングルタスク上で動作するアプリケーションでマルチコネクションを確立し、管理することが可能となる。

【0108】第3の実施の形態

第3の実施の形態は受信バッファ5aの使用量を定期的に検知して許容コネクション数を変化させ、処理飽和状態からのリカバリーを迅速にするとともに、低要求コネクション確立時にも、効率良く動作させるようにしたものである。

【0109】図11は第3の実施の形態によるインターバルタイマ処理の要部を示すブロック図、図12は許容コネクションを増減させる際に使用するテーブルである。

【0110】まず、図5に示したタイマ処理手段31に受信バッファ5aの使用量をセンスするバッファ数センス部50と、図12に示したテーブル51aを参照して処理負荷を判断する判断部51と、許容コネクション数を設定するコネクション数設定部52とを設ける。

【0111】次に動作について図13に従って説明する。図13は第3の実施の形態の動作を示すフローチャートである。ステップS1でバッファ数センス部50はインターバルタイマ21の周期で受信バッファ5aの使用量をセンスし、ステップS2で受信バッファ5aの使用量をグローバルに確保したメモリ5の記憶領域Xに格納する。また、1回目のセンス時は、「前回のセンス時

の受信バッファ使用量 (Y) 」データは存在しないので c、メモリ 5 の記憶領域 Y を 0x F F F F にしておく。

【0112】ステップ S3 で記憶領域 Y の内容が 0x F F F F か否かをチェックし、0x F F F F の場合にはステップ S5 に分岐し、否の場合はステップ S4 に分岐する。ステップ S4 で「前回のセンス時の受信バッファ使用量 (Y) 」データが存在するので、「今回のセンス時の受信バッファ使用量 (X) 」を用い、図 12 に示すテーブル 51 a との比較により、許容コネクション数の変更量を決定し、メモリ 5 の記憶領域 Z へ書き込む。

【0113】ステップ S5 で、次のセンスに備え、「今回センス時の受信バッファ使用量 (X) 」を「前回のセンス時の受信バッファ使用量 (Y) 」へコピーする。

【0114】ステップ S6 で、記憶領域 Z の内容が 0 か否かをチェックして、0 ならばステップ S1 に戻って次のインターバルタイマ 21 による起動を待ち、否ならばステップ S7 で記憶領域 Z の内容に応じたコネクション数を増減させる。コネクション数設定部 52 は、記憶領域 Z の内容が 0 以外の時に起動し、記憶領域 Z の内容に応じたコネクション数を増減させる。本処理は、記憶領域 Z の内容が正の数である時には、各コネクションに必要な領域の確保とイニシャライズ行う。記憶領域 Z の内容が負の数である時には各コネクションに必要な領域を使用不可能にする。許容数までコネクションが接続している場合は、コネクションが切断され次第各コネクションに必要な領域を使用不可能にする。

【0115】図 12 を用いてコネクション数設定部の動作を説明する。コネクション数設定部 52 は、記憶領域 Z の内容が 0 以外の時に起動し、記憶領域 Z の内容に応じたコネクション数を増減させる。記憶領域 Z の内容が正の数である時には、各コネクションに必要な領域の確保とイニシャライズ行う。記憶領域 Z の内容が負の数である時には各コネクションに必要な領域を使用不可能にする。許容数までコネクションが接続している場合は、コネクションが切断され次第各コネクションに必要な領域を使用不可能にする。

【0116】現在及び前回の使用受信バッファ量が全受信バッファ量の 20% 以下であった場合は、要求負荷の低いコネクションが全コネクションの大半を占めている可能性が高いので、コネクションの許容数を 1 つ増加させる。

【0117】また、現在の使用受信バッファ量が全受信バッファ量の 80% 以上であった場合には、後工程の処理の負荷が高く、受信バッファのオーバフローが予想されるのでコネクションの許容数を 1 つ減少させる。

【0118】現在の使用受信バッファ量が全受信バッファ量の 21% ~ 50% の間で、前回の使用受信バッファ量が 51% 以上であった場合は、処理負荷が下降していることを示しているので、急激な処理負荷の下降に備えコネクションの許容数を 1 つ増加させる。

【0119】現在の使用受信バッファ量が全受信バッファ量の 51% ~ 80% の間で、前回の使用受信バッファ量が 50% 未満であった場合は、処理負荷が上昇していることを示しているので、急激な処理負荷の上昇に備えコネクションの許容数を 1 つ減少させる。

【0120】第 3 の実施の形態によれば、現在行っている処理の重さにより確立するコネクション数を変化させることにより、処理飽和状態からのリカバリーを早くし低要求コネクション確立時にも、より効率良く動作させることができる。

【0121】第 4 の実施の形態

第 4 の実施の形態は、図 1 に示した印刷部 10 の展開タスク 10 a 等からの高負荷通知に対し、インターバルタイマ 21 のタイマ間隔を可変とすることにより、展開タスク 10 a 等のプライオリティの低いタスクが処理負荷の高い状態にある時に、I F C タスク 22 等のプライオリティの高いタスクを制御できるようにしたものである。

【0122】図 14 は第 4 の実施の形態による負荷監視処理のブロック図である。負荷監視部 60 は図 5 に示した展開タスク 10 a 等の処理の負荷を監視するためにプライオリティの低いタスク内に設けられている。負荷通知エリア 61 はメモリ 5 のグローバルエリアに設けられ、負荷監視部 60 から通知を受信し、高負荷時に他の処理へそれを通知する。タイマ間隔値変更手段 21 c はインターバルタイマ 21 に設けられ、高負荷時にインターバルタイマ 21 のカウンタ 21 b のタイマ間隔値を変更する。

【0123】図 15 はプリンタにおける展開処理とエンジン処理等とのタイムチャートであり、(a) ~ (d) は展開処理、エンジン処理、タイマ処理、その他の処理を示している。

【0124】時間 W は、エンジン処理期間 T1 間に展開処理が動作できる時間を示し、この時間内に一定のデータの展開が終了しない場合にオーバランが発生する。タイマ処理と、その他の処理（共に展開処理よりもプライオリティは高い）が時刻 t1、t2、t4、t5 で発生すると、実際の展開処理動作は、矢印で示す範囲となり、オーバランが発生しやすくなる。

【0125】図 16 は第 4 の実施の形態の動作を示すフローチャートである。

【0126】次に動作について図 16 のステップに従って説明する。ステップ S1 でインターバルタイマ 21 のタイマ処理 21 a は、タイマ割込み毎にカウントするカウンタ 21 b がタイマ間隔値 T 以上になったかをセンスし、以下だった場合はステップ S2 でカウンタ 21 b をインクリメントし、以上だった場合はステップ S3 でプロトコルスタック内のタイマ処理手段 31 を呼出するために I F C タスク 22 に対してメッセージ (message-S) を出力する。ステップ S4 でカウンタ 21 b は 0 にリセット

トされる。

【0127】ステップS5でタイマ間隔値変更手段21cは負荷通知エリア61を参照し、負荷監視部60からの高負荷通知が受信されていると、ステップS8でカウンタ21bのタイマ間隔値Tを2倍にのぼす。高負荷通知が受信されていない場合には、ステップS6でカウンタ21bのタイマ間隔値Tを調べ、変更前のデフォルト値であった場合には本処理を終了する。変更が入っている場合には、ステップS7でタイマ間隔値Tを半分に短縮する。負荷監視部60から負荷通知エリア61に高負荷通知が出力され続けていると、タイマ処理手段31からポストされて起動されるプロトコルスタック26中のタイマ処理は2倍単位でその間隔が延び、負荷通知エリア61に高負荷通知が出力されていない時間が続くと、デフォルトのタイマ間隔値まで二分の一単位で間隔は短縮される。

【0128】本実施の形態をプリンタに適用し、展開タスク等からの高負荷通知に対し、タイマ処理の間隔を可変とすることにより、図15の時間Wに示す間隔内のタイマ処理時間を少なくすることが可能となる。また、図14に示す負荷通知エリアをグローバル領域に設定することにより、タイマ処理以外の高負荷処理へも同時に通知することができ、対策を入れることが可能となる。

【0129】第4の実施の形態によれば、展開タスク等からの高負荷通知に対し、タイマ処理の間隔を可変とすることにより、展開タスク等のプライオリティの低いタスクが処理負荷の高い状態にある時に、I F C等のプライオリティの高いタスクを制御できる。このことにより、プリンタではオーバーランや、オーバーフロー等の性能を向上させ、ネットワーク処理の集中に関係なく安定した性能を保つことができる。

【0130】第5の実施の形態

第1の実施の形態から第4の実施の形態までは単独のプリンタ記述言語を備えたプリンタに対応したが、第5の実施の形態はプリンタが複数のプリンタ記述言語、プリンタコントロールラングエジ (PRINTER CONTROL LANGUAGE: 以後PCLと記す)、ポストスクリプト (POST SCRIPT: 以後PSと記す)、プリンタジョブラングエジ (PRINTER JOB LANGUAGE: 以後PJLと記す) 等を備えた場合に対応できるようにしたものである。

【0131】図17は第5の実施の形態によるプリンタのネットワークアーキテクチャを示すブロック図であり、第2の実施の形態によるプリンタと異なるところは、PCL、PS、PJL等のプリンタ記述言語に対応するI F Cタスク40から共通に使用できるようにネットワークドライバ25を再入可能 (一つの関数からコールされ、リターンする前に他の関数からコールすることが可能である仕様) にした点と、セントロニクス、RS-232C等のインターフェースをネットワークドライバ25と同様に共通関数にした点である。

【0132】プリンタ言語判別部41はプリンタ記述言語に対応する各I F Cタスク40やセントロニクス、RS-232C等のインターフェースと、各プリンタ記述言語に対応する編集部、例えばPCL用編集部42、PS用編集部43に接続され、各I F Cタスク40から出力されるデータのプリンタ記述言語を判別して対応する編集部に出力している。

【0133】図18はI F Cタスクの構造を示すブロック図であり、第2の実施の形態によるI F Cタスクの構造と異なるところは、ネットワークドライバ25がI F Cタスク40の外にある点と、上位層から呼ばれる共通化ドライバ関数44をI F Cタスク40の内部に設けた点である。

【0134】図19は共通関数部の構造を示すブロック図であり、I F Cタスク40のアクセスを制御するドライバ (Ethernet方式、IEEE802.2方式、IEEE802.3方式、SNAP方式等) をまとめたネットワークドライバ25と、ネットワーク接続用H/Wを制御するドライバをまとめたNICドライバ45とからなる。

【0135】次に動作を図18を参照して説明する。第2の実施の形態と異なるところは、メッセージ処理手段24がNICドライバ45からなる受信割り込み手段20から矢印(a)に示すメッセージを受信すると、I F Cタスク40の外にあるネットワークドライバ25を関数コールする点であり、その他の動作は第2の実施の形態と同様である。

【0136】第5の実施の形態によれば、メモリ容量の大きいネットワークドライバをPCL、PS、PJL等のプリンタ記述言語に対応するI F Cタスクに設けることなく、各I F Cタスクから共通に使用できるようにしたので、メモリを効率よく使用できる。

【0137】第6の実施の形態

第6の実施の形態によるプリンタのネットワークアーキテクチャは、第5の実施の形態と同じである。メッセージ処理手段24は各処理手段からのメッセージを受信するイベントコントロールブロック (Event Control Block: 以後ECBと記す) からメッセージを取り出して処理するが、ECBが複数あって、処理する優先度をECB間に設けてあればタスク内の処理にも優先度を設けることができ、効率良くCPU1を動作させることができるが、ECBが一つの場合には、タスク内の処理に優先度を設けることができないのでCPU1を効率良く動作できなくなる。

【0138】第6の実施の形態はECBが一つの場合にも効率の良い動作ができるようにしたものである。

【0139】図20は第6の実施の形態によるメッセージ処理手段の動作を示すブロック図である。メッセージ処理手段24はECB46とソケット処理手段28の上位側待ち行列WAPFQueue 47とを参照し、ECB46からメッセージを取り出してプロトコルスタック26

ヘブプロトコルを処理するように要求するか、上位側待ち行列WAPFQueue 47の処理をコールするかを判定する。

【0140】図21は図20に示したメッセージ処理手段の動作を示すフローチャートである。メッセージ処理手段24はステップS1でソケット処理手段28に上位層待ち行列WAPFQueue 47が生成されてあるか否かをチェックして生成されてあるならばステップS2に分歧し、否ならばステップS4に分歧する。

【0141】ステップS2で上位層待ち行列数が所定量、例えば10個未満か否かをチェックして10個未満ならばステップS3に分歧し、否ならばステップS5に分歧する。ステップS3でECB46にメッセージが受信してあるか否かをチェックして受信してあるならばステップS4に分歧し、否ならばステップS5に分歧する。ステップS4でプロトコルスタック26ヘブプロトコルを処理するように要求し、ステップS5でソケット(High)28bをコールする。

【0142】尚、本実施の形態は第2の実施の形態にも適用できる。

【0143】第6の実施の形態によれば、OSからポストされたメッセージを格納するECBを一つのタスクに複数設けられない場合にも、タスク内の処理に優先度を設けることができるので、CPUを効率良く動作させることができる。

【0144】第7の実施の形態

第7の実施の形態によるプリンタのネットワークアーキテクチャは、第5の実施の形態と同じである。ジョブ制御モジュール29はデータのクライアント識別子を格納する共通テーブルを有し、データを出力する際にクライアント識別子を格納し、上位層からデータ送信要求を受ける際には共通テーブルに基づいてクライアント識別子を解析し、同一コネクションを通じてパケットを送受信するようにしたものである。

【0145】図22は第7の実施の形態を示すブロック図である。2のパケット処理手段30はジョブ制御モジュール29にデータのクライアント識別子(以後クライアントIDと記す)等を登録する共通テーブルを有する。

【0146】共通テーブルにはデータを印刷部の編集タスクに送信する際に使用するDMD48と、プリンタ言語判別部41からデータ送信要求を受ける際に登録するSendDescriptor 49とがある。例えば、ネットワークドライバ25を経由して3つのパケットを受信したとする。各パケットはプロトコルを処理されてアプリケーションAP1、AP2、AP4を通じてジョブ制御モジュール29に送信されてくるものとする。

【0147】クライアントID C1、C2、C3は受信割り込み手段20がパケットを受信した際にDMD48に登録される。ジョブ制御モジュール29はアプリケ

ーションAP1、AP2、AP4から受信する印刷データとともにクライアントID C1、C2、C3をプリンタ言語判別部41に送信する。クライアントIDは32ビットで構成される。最初の8ビットはIFCタスク40、セントロニクス、RS-232C等のインターフェースの種別を示し、次の24ビットは各インターフェースによって固有となる。例えばIFCタスク40の場合には、8ビットがアプリケーションを示すID(AP1、AP2、...)で、残りの16ビットが各アプリケーションにおける通番となる。このフォーマットにより同一のアプリケーションにクライアントの接続が集中しても対応することができる。

【0148】プリンタ言語判別部41は受信したクライアントに対してデータを送信する場合にはインターフェースの種別をクライアントIDの最初の8ビットから判断し、そのインターフェースのSend Descriptor 49に登録する。

【0149】ジョブ制御モジュール29はSend Descriptor 49を参照してアプリケーションを識別するビットを読み取り、対応するアプリケーションをコールする。

【0150】尚、本実施の形態は第2の実施の形態にも適用できる。

【0151】第7の実施の形態によれば、IFCタスク等のインターフェースを使用してパケットの授受を行う上位層が、クライアントIDの中身を意識することなく応答パケットの送信先を特定するので、要求するクライアントに正確に応答パケットを送信できる。

【0152】第8の実施の形態

第8の実施の形態によるプリンタのネットワークアーキテクチャは、第5の実施の形態と同じである。第8の実施の形態はプリンタ等に組み込まれた受信バッファのサイズが小さい場合に、効率的な受信バッファ空き容量を算出して受信速度を向上させるようにしたものである。

【0153】図23はプリンタ言語判別部41とジョブ制御モジュール29との間のデータ受け渡し処理を示す説明図、図24は送信側TCPと受信側TCPとの間のデータ受け渡し処理を示す説明図、図25は第8の実施の形態を示すブロック図である。第7の実施の形態で説明したように、プリンタ言語判別部41は処理状況に応じて受信データをジョブ制御モジュール29に要求するが、図23に示すように、データ要求のポーリング間隔が常に一定に限らない。

【0154】また、TCP間の受信処理では、図24に示すように、受信パケットを複数受信して一つの応答パケットを返すブロック転送方式を使用している。さらに、トランスポート層の機能を使用するなど、送受信端末間の距離制限は事実上無いに等しい。従って、パケット転送時間Tは非常に短い場合もあるが、非常に長い場合もある。

【0155】そこでジョブ制御モジュール29に定期的

にポーリング間隔を監視して、ジョブ制御モジュール29からプリンタ言語判別部41に転送される単位時間内のデータ数Rを計数する転送データ計数手段50を設ける。

【0156】また、TCPの受信パケット処理時、パケット転送時間Tを算出するパケット転送時間更新手段51をプロトコルスタック26に設ける。

【0157】また、ジョブ制御モジュール29には、パケット転送時間Tと単位時間にデータを上位層に転送する転送データ数Rと現在の受信バッファ空容量Wとから次のパケットブロック転送において受信可能なデータ容量Xを算出する受信可能データ容量算出手段52を設ける。

【0158】受信可能なデータ容量Xは、

$$X = (R \times T + W) \dots \dots \dots (14)$$
 で算出される。

【0159】メモリ5にはパケット転送時間Tを格納する格納エリア53、NICドライバ45が受信パケットを受信する毎に更新される受信バッファ空容量Wを格納する格納エリア54、受信可能データ容量算出手段52が算出した受信可能なデータ容量Xを格納する格納エリア55が設けてある。

【0160】プロトコルスタック26の応答パケット作成手段56は応答パケットを作成する際に格納エリア55の内容を応答パケットに盛り込んで送信する。

【0161】尚、本実施の形態は第2の実施の形態にも適用できる。

【0162】第8の実施の形態によれば、実際の受信バッファサイズにとらわれず、後工程の処理速度と前工程の受信速度を接近させることにより、高速で効率的な受信を実現できる。

【0163】第9の実施の形態

第9の実施の形態によるプリンタのネットワークアーキテクチャは、第5の実施の形態と同じである。第9の実施の形態は、受信バッファが受信データで満杯になり、送信側からの接続を確認するポーリングパケットが受信できず、そのパケットに対する応答パケットを送信できない状態が発生する。その状態が長時間続いた場合にも接続が切断されないようにしたものである。

【0164】図26は第9の実施の形態を示すブロック図であり、図19に示したデータリンク層以下を処理するNICドライバの受信構造を示す。NICドライバ45は、第7の実施の形態で説明した通り、受信パケットを自層及びその上位層で共通管理する共通テーブルにリンクされている。共通テーブルは各層が処理した情報や自層の後工程である層に対して通知する情報を格納する領域としてメモリ5に設けてある。

【0165】受信バッファ5aには常時空けておく「常時空きエリア」と、常時使用する「常時使用エリア」とを設けておく。NICドライバ45は「常時空きエリ

ア」と「常時使用エリア」とを認識し、受信パケットを優先的に「常時使用エリア」に受信するとともに共通テーブルに情報（受信データサイズ、受信データ格納領域の先頭アドレス、等）を格納する。「常時使用エリア」が受信パケットで満たされると、「常時空きエリア」に受信できるように制御する。

【0166】NICドライバ45は「常時空きエリア」に受信している状態でも、受信パケットの処理が進み、「常時使用エリア」に空き領域ができた場合には、以降「常時使用エリア」に受信するように制御する。「常時空きエリア」は1～2パケット分の領域からなり、このエリアにパケットを受信した場合には「常時空きエリア」にリンクした共通テーブル（特）にその情報を格納する。

【0167】図27は第9の実施の形態の動作を示すフローチャートであり、(A)はNICドライバの動作を示し、(B)はTCPバイパス処理の動作を示す。

【0168】まず、NICドライバ45の動作を説明する。ステップS1で「常時空きエリア」にパケットが受信されたか否かをチェックし、受信ならばステップS2に分岐し、否ならばステップS5に分岐する。ステップS2で共通テーブル（特）に空きがあるか否かをチェックし、空きがあるならばステップS3に分岐し、否ならばステップS6に分岐する。ステップS3で「常時空きエリア」に受信されたパケットの情報を共通テーブル（特）に格納する。

【0169】ステップS4でTCPバイパス処理を呼び出し、TCPバイパス処理後、ステップS1に戻る。ステップS5では通常の処理を行ってステップS1に戻る。ステップS6では「常時空きエリア」に受信したパケットを破棄してステップS1に戻る。

【0170】ステップS7で受信パケットがパケット送信元からのポーリングパケットか否かをチェックし、ポーリングパケットならばステップS8に分岐し、否ならばステップS9に分岐する。受信側のTCPは受信バッファが満杯になった等の理由で一時的に受信を止めたい場合には、応答パケットの所定のエリアに0を格納して送信側のTCPに送信する。送信側のTCPはこの応答パケットを受信すると、パケットの送信を停止する。そして、パケットの送信を開始するタイミングを獲得するために、受信側のTCPにポーリングパケットを定期的に送信する。

【0171】ステップS8で応答パケットを作成して送信する。ステップS9で「常時空きエリア」に受信したパケットを破棄する。

【0172】尚、本実施の形態は第2の実施の形態にも適用できる。

【0173】第9の実施の形態によれば、受信バッファが受信データで満杯になり、長時間受信バッファが空かない場合でも、送信側からの接続を確認するポーリング

パケットが受信でき、そのパケットに対する応答パケットを送信できるので、コネクションを維持できるとともに、その間低負荷の処理により接続を保つことができる。

【0174】

【発明の効果】本発明は、以上説明したように構成されているので以下に記載される効果を奏する。ネットワーク層、トランスポート層を受信パケット一括処理手段という一つの機能で処理するようにしたので、プロトコル処理を1回の起動で、受信バッファに受信された全パケットを処理できるので、OSのオーバーヘッド量を減少させることができる。

【0175】また、ネットワーク層、トランスポート層を一括処理することにより、プロトコルをネットワーク層、トランスポート層別に処理する場合に比べてメモリ上の作業領域を削減できる。

【0176】さらに、それぞれのコネクションに属する最後の受信パケットに対する応答パケットを割込み中にネットワークに一括して送信する送信パケット一括処理手段を備えたことにより、送信処理時間を大幅に短縮でき、その分他の処理にCPUを動作させることができるので、効率良く動作するネットワークプロトコルのインプリメント手法を提供できる。

【0177】同様に、少なくともネットワーク層、トランスポート層を第1のパケット処理手段という一つの機能で処理するようにし、セッション層、プレゼンテーション層、アプリケーション層を第2のパケット処理手段という一つの機能で処理するようにして、その2つの機能間にソケット処理手段及びメッセージ処理手段を介在させて処理の優先度を持たせるようにしたことにより、OSのオーバーヘッド量を減少させることができる。

【0178】また、ネットワーク層、トランスポート層を一括処理することにより、プロトコルをネットワーク層、トランスポート層別に処理する場合に比べてメモリ上の作業領域を削減できる。

【0179】また、低機能OSを用いて他のインターフェース処理をも受け付けるできるようにしたので、同一タスク上で他のホストインタフェース(セントロニクス、RS-232C)との共存を可能にできる。

【図面の簡単な説明】

【図1】第1の実施の形態によるプリンタのネットワークアーキテクチャを示すブロック図である。

【図2】第1の実施の形態によるネットワークの構成図である。

【図3】図1に示したネットワークアーキテクチャのタイムチャートである。

【図4】従来と本発明との送信処理の比較を示すタイムチャートである。

【図5】第2の実施の形態によるプリンタのネットワークアーキテクチャを示すブロック図である。

【図6】アプリケーションFTPの構成を示すブロック図である。

【図7】コネクション管理テーブルの説明図である。

【図8】コネクションの状態と設定値との関係を示す説明図である。

【図9】処理ステップ数と負荷との関係を示す負荷算出テーブルである。

【図10】パケットを再送する際の処理を示すフローチャートである。

【図11】第3の実施の形態の要部を示すブロック図である。

【図12】許容コネクションを増減させる際に使用するテーブルである。

【図13】第3の実施の形態の動作を示すフローチャートである。

【図14】第4の実施の形態による負荷監視処理のブロック図である。

【図15】プリンタにおける展開処理とエンジン処理等とのタイムチャートである。

【図16】第4の実施の形態の動作を示すフローチャートである。

【図17】第5の実施の形態によるプリンタのネットワークアーキテクチャを示すブロック図である。

【図18】IFCタスクの構造を示すブロック図である。

【図19】共通関数部の構造を示すブロック図である。

【図20】第6の実施の形態を示すブロック図である。

【図21】メッセージ処理手段の動作を示すフローチャートである。

【図22】第7の実施の形態を示すブロック図である。

【図23】プリンタ言語判別部とジョブ制御モジュールとの間のデータ受け渡し処理の説明図である。

【図24】TCP間のデータ受け渡し処理の説明図である。

【図25】第8の実施の形態を示すブロック図である。

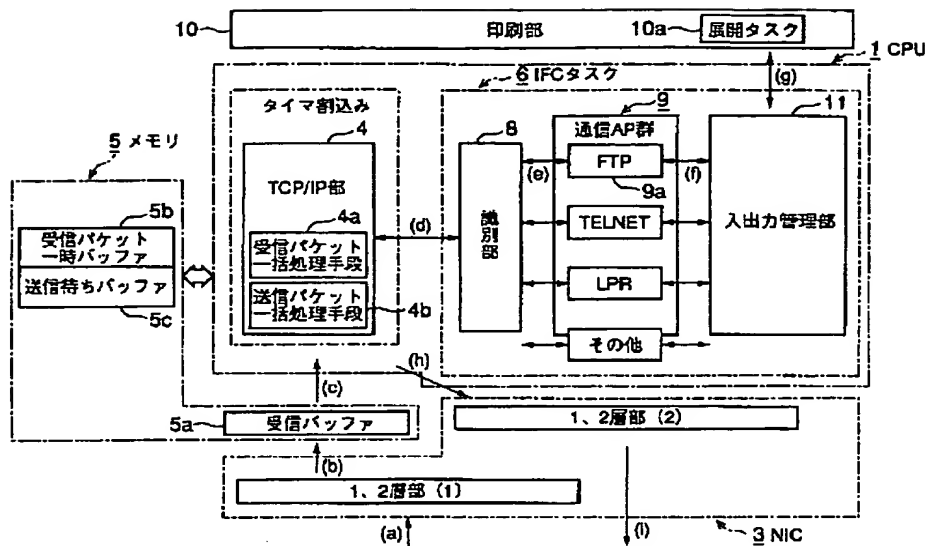
【図26】第9の実施の形態を示すブロック図である。

【図27】第9の実施の形態の動作を示すフローチャートである。

【符号の説明】

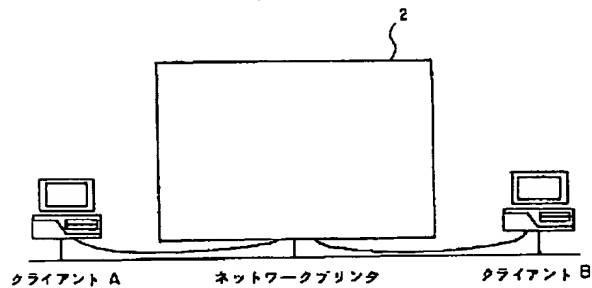
- 1 CPU
- 3 NIC
- 4 TCP/IP部
- 5 メモリ
- 6、22、40 IFCタスク
- 23 第1のパケット処理手段
- 30 第2のパケット処理手段

【図1】



第1の実施の形態によるプリンタのネットワークアーキテクチャを示すブロック図

【図2】



第1の実施の形態によるネットワークの構成図

【図7】

コネクション1	コネクション2	コネクション3	コネクションn
コネクションの状態	コネクションの状態	コネクションの状態	コネクションの状態
処理中のコマンド	処理中のコマンド	処理中のコマンド	処理中のコマンド
コネクションNo.	コネクションNo.	コネクションNo.	コネクションNo.
送信パケット数	送信パケット数	送信パケット数	送信パケット数
相手先ポートNo.	相手先ポートNo.	相手先ポートNo.	相手先ポートNo.

コネクション管理テーブルの説明図

【図3】

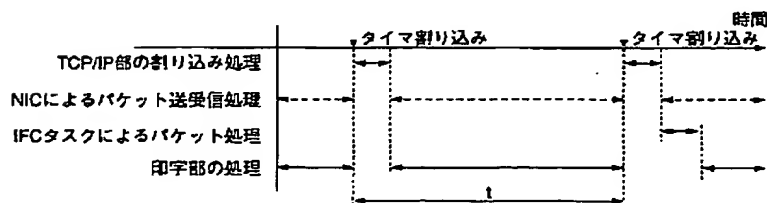
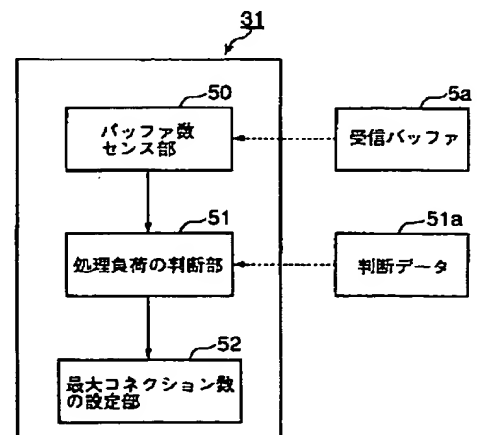


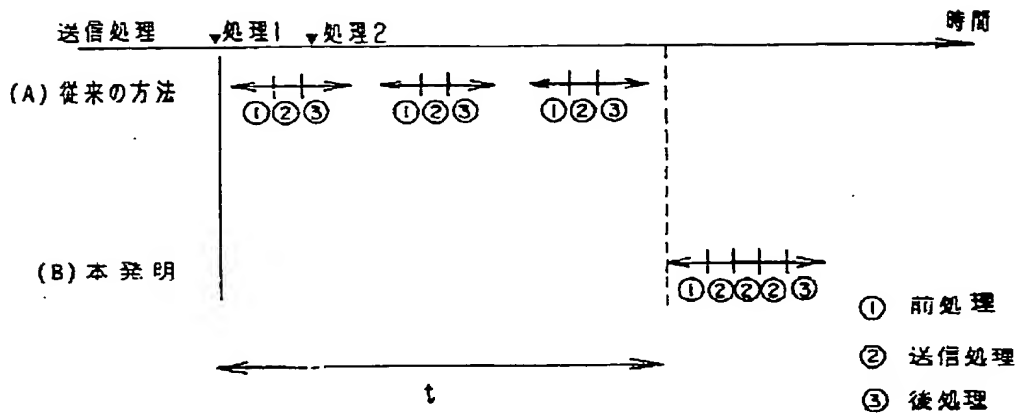
図1に示したネットワークアーキテクチャのタイムチャート

【図11】



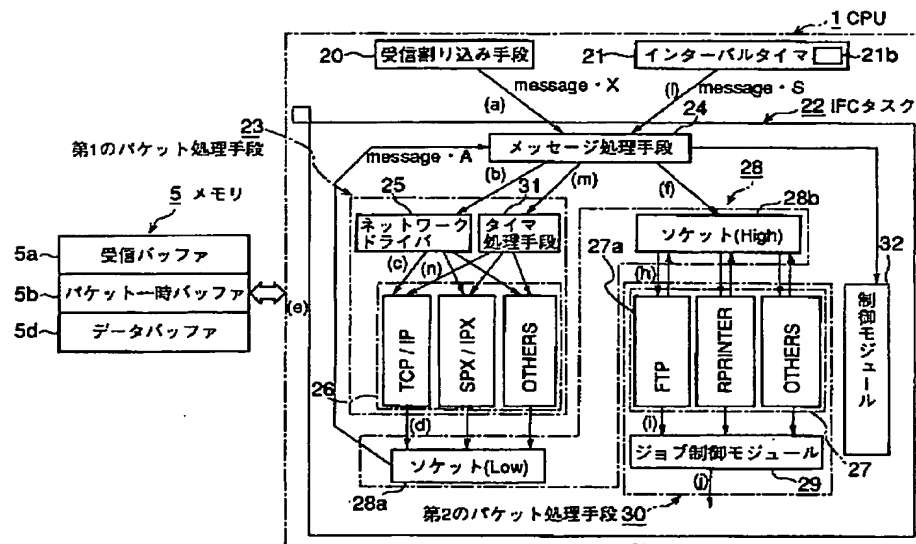
第3の実施の形態の要部を示すブロック図

【図4】



従来と本発明との送信処理の比較を示すタイムチャート

【図5】



第2の実施の形態によるプリンタのネットワークアーキテクチャを示すブロック図

【図8】

コネクションの状態	設定値
コネクションが確立していない状態	0
初期コネクションが確立した状態	1
初期コネクションからログインを許可した状態	2
データコネクションを確立要求中の状態	3
データコネクションが確立した状態	4

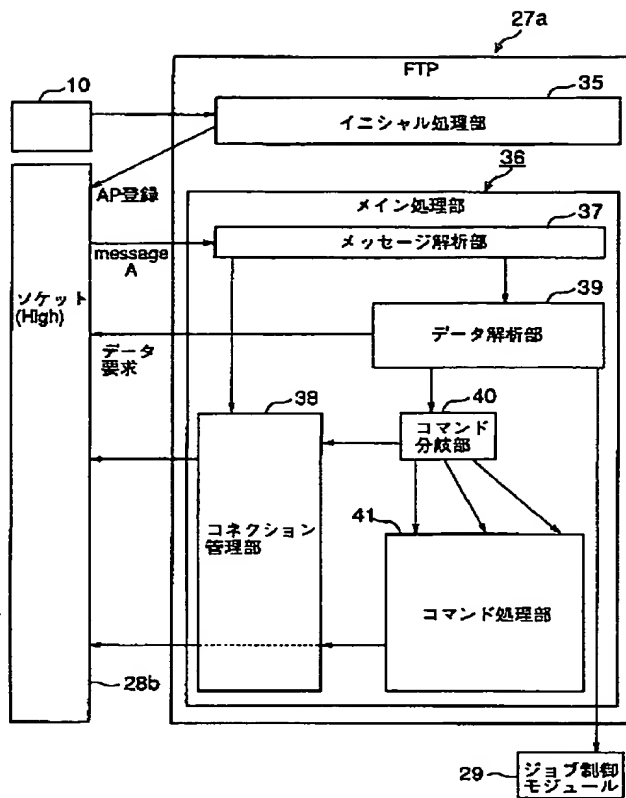
コネクションの状態と設定値との関係を示す説明図

【図9】

	TCP/IP		SPX/IPX		Others	
	STEP数	G	STEP数	G	STEP数	G
処理1	70	70	40	40	なし	0
処理2	120	120	113	113	130	130
処理3	120	120	170	170	178	178
処理4	なし	0	70	70	104	104

処理ステップ数と負荷との関係を示す負荷算出テーブル

【図6】



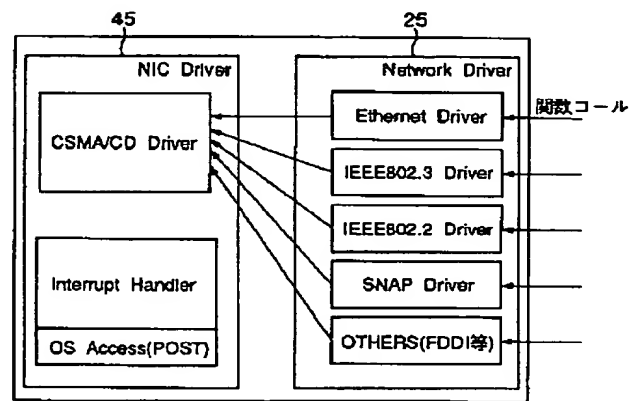
アプリケーションFTPの構成を示すブロック図

【図12】

現在の使用受信 バッファ量	前回センサ時の 使用受信バッファ量	コネクション数の変更量
20%以下	20%以下	1増加
21~50%	51%以上	1増加
50~80%	50%未満	1減少
80%以上	-	1減少

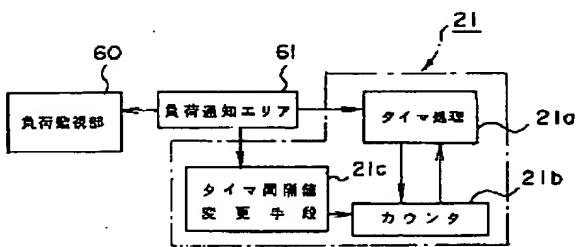
許容コネクションを増減させる際使用するテーブル

【図19】



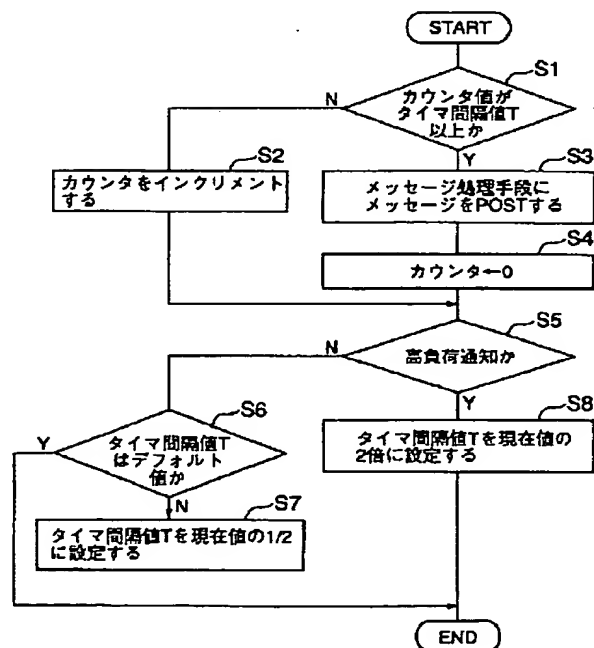
共通関数部の構造を示すブロック図

【図14】



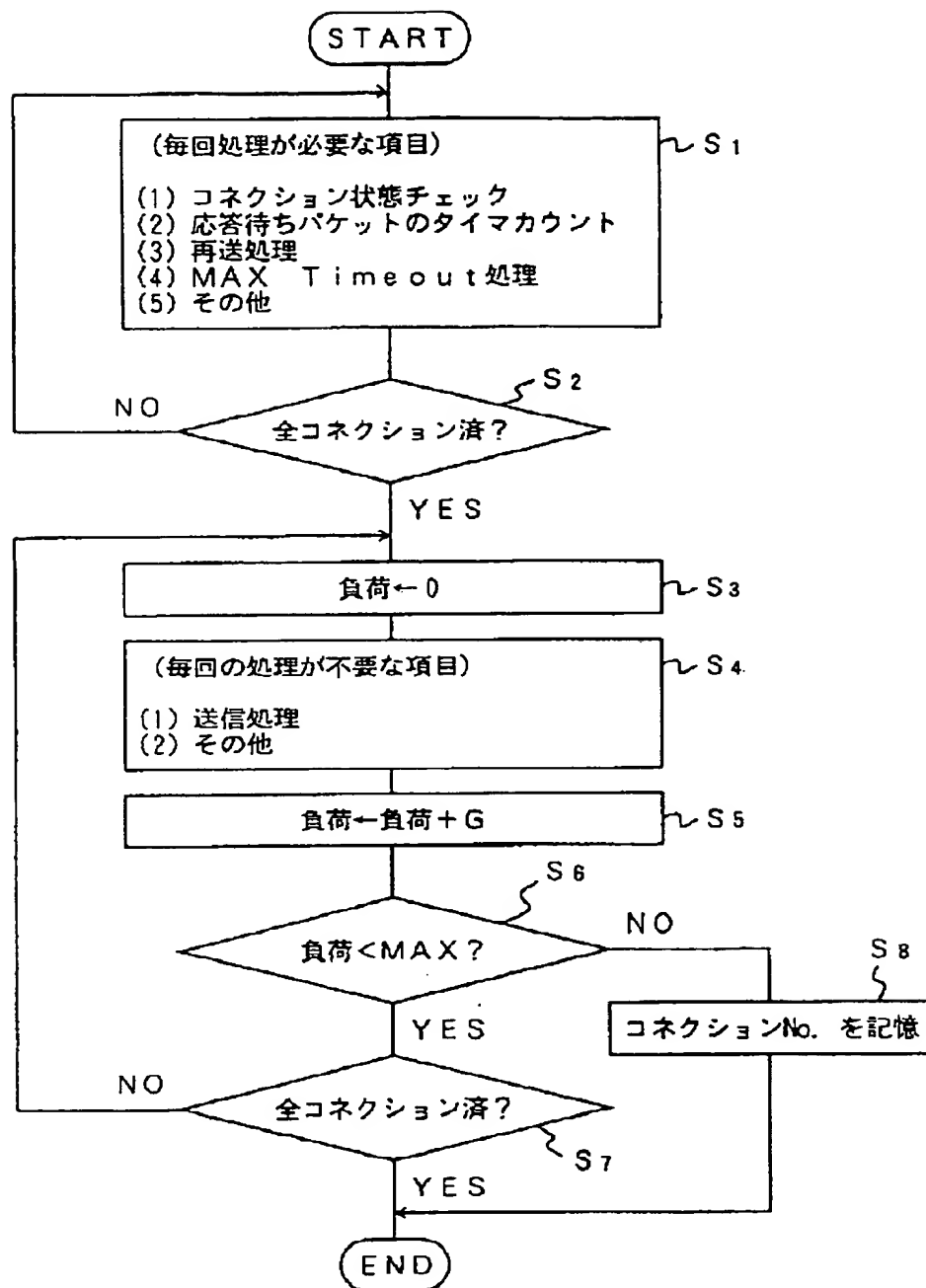
第4の実施の形態による負荷監視処理のブロック図

【図16】



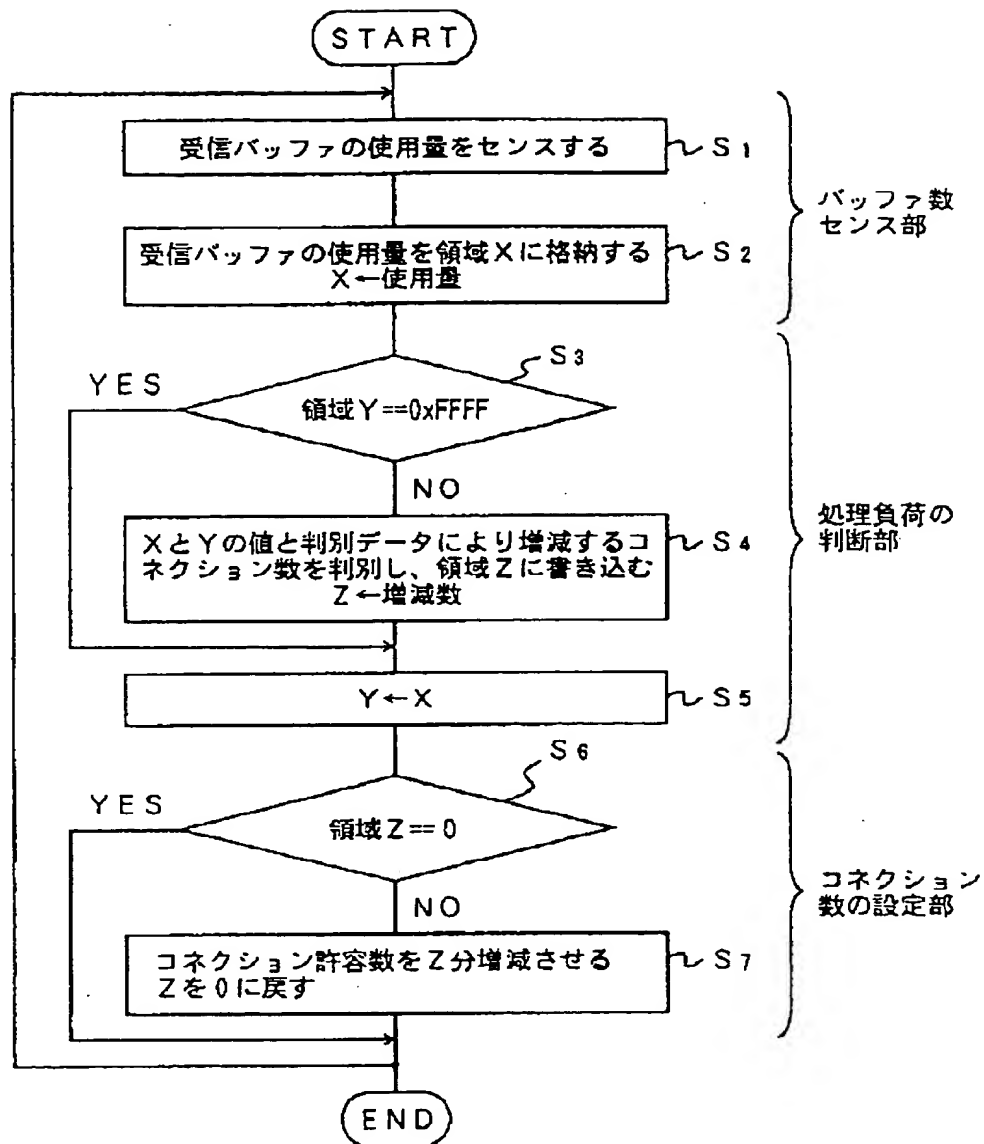
第4の実施の形態の動作を示すフローチャート

【図10】



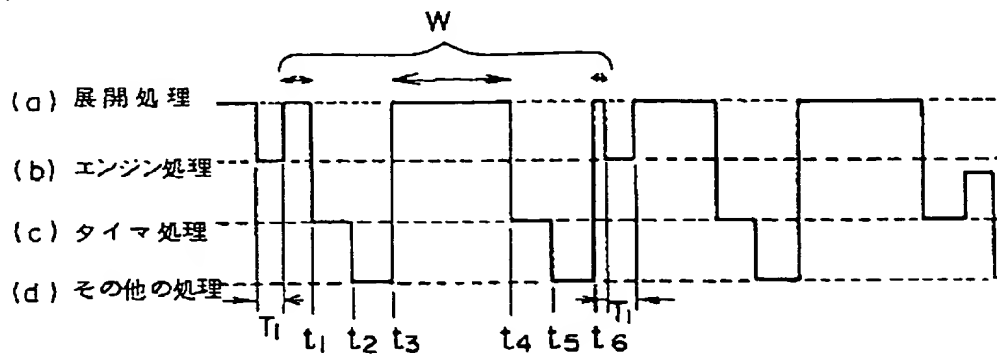
パケットを再送する際の処理を示すフローチャート

【図13】



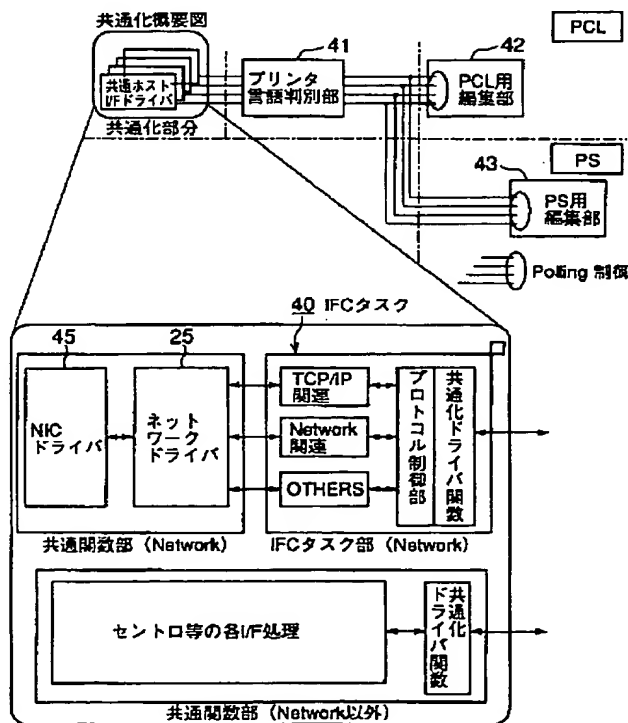
第3の実施の形態の動作を示すフローチャート

【図15】

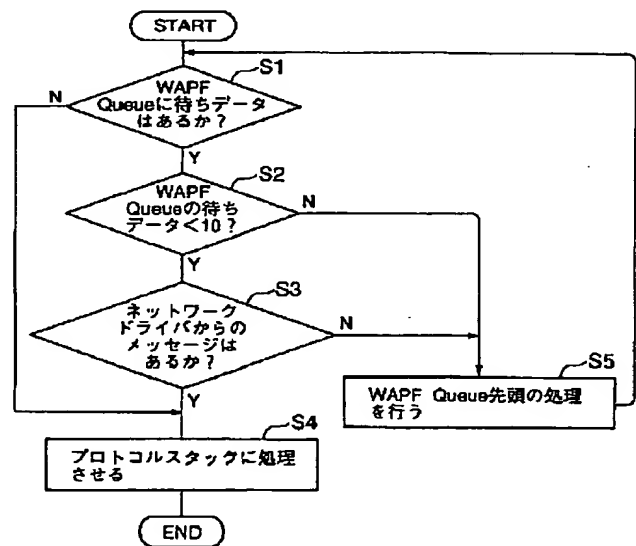


プリンタにおける展開処理とエンジン処理等とのタイムチャート

【図17】



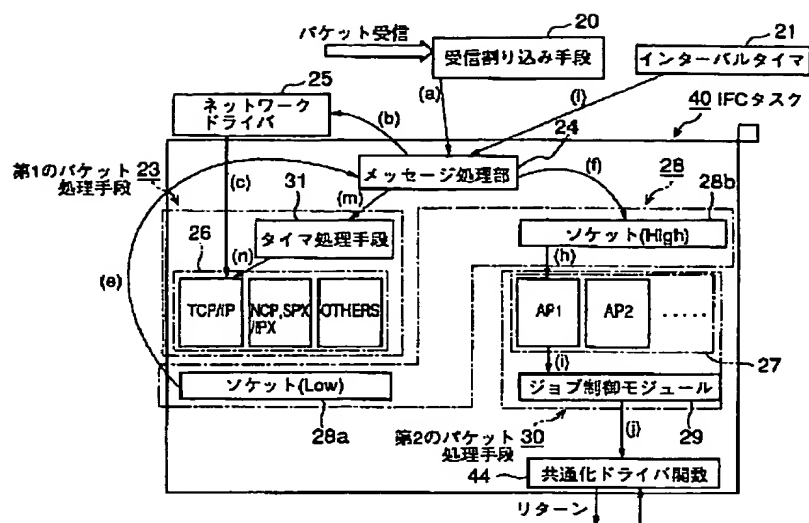
【図21】



メッセージ処理手段の動作を示すフローチャート

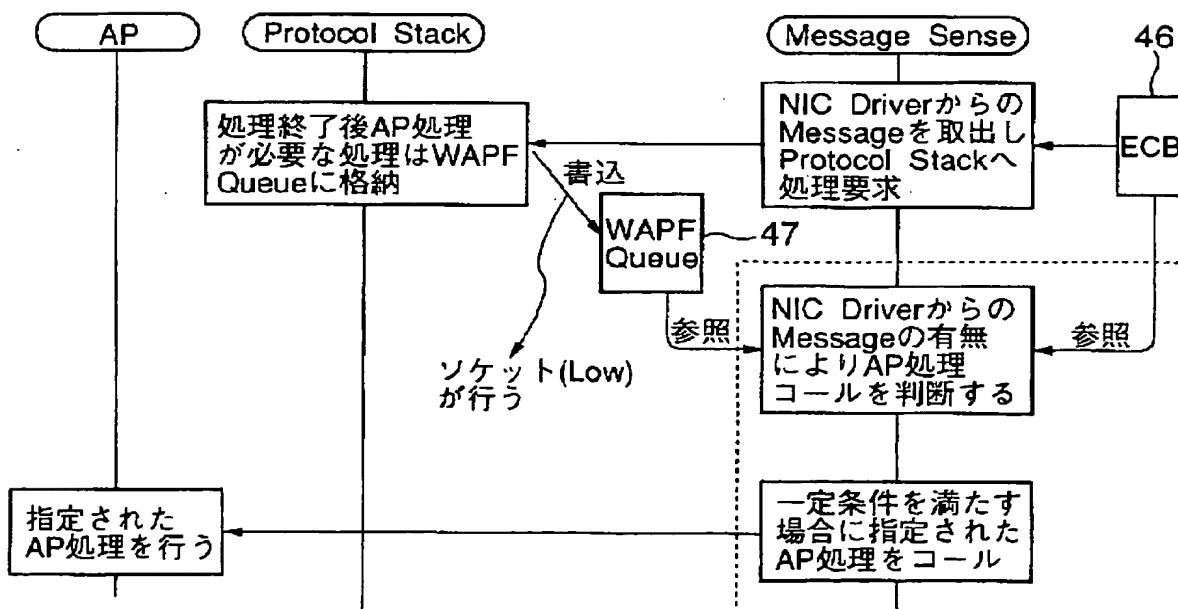
第5の実施の形態によるプリンタのネットワークアーキテクチャを示すブロック図

【図18】



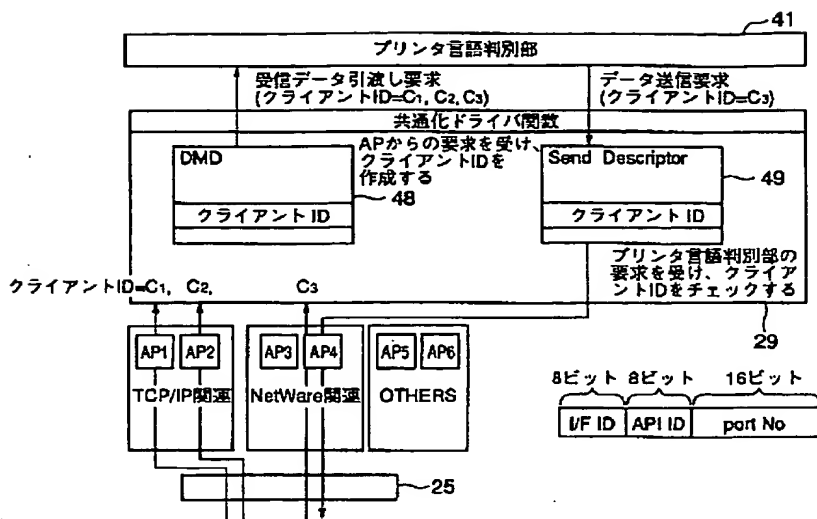
IFCタスクの構造を示すブロック図

【図20】



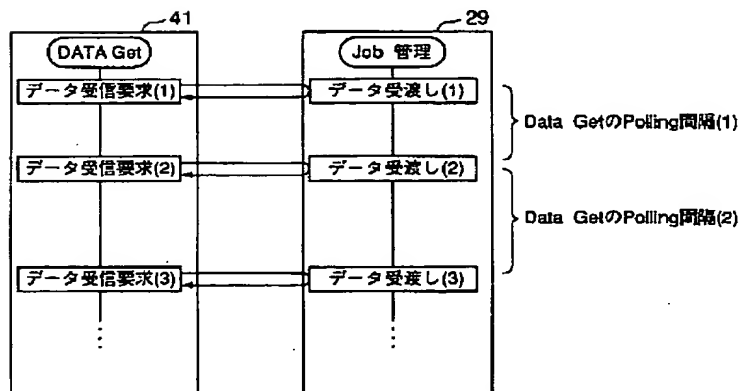
第6の実施の形態を示すブロック図

【图 2 2】



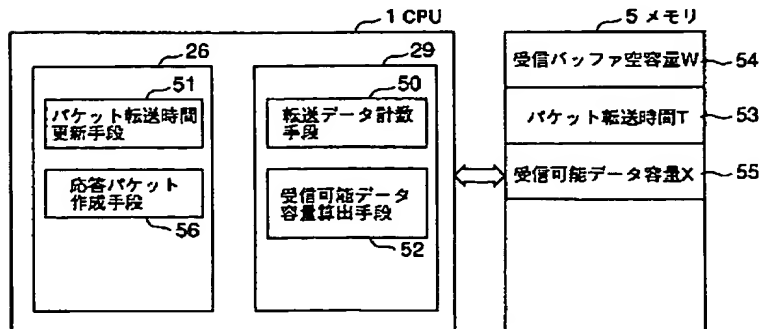
第7の実施の形態を示すブロック図

【图 2 3】



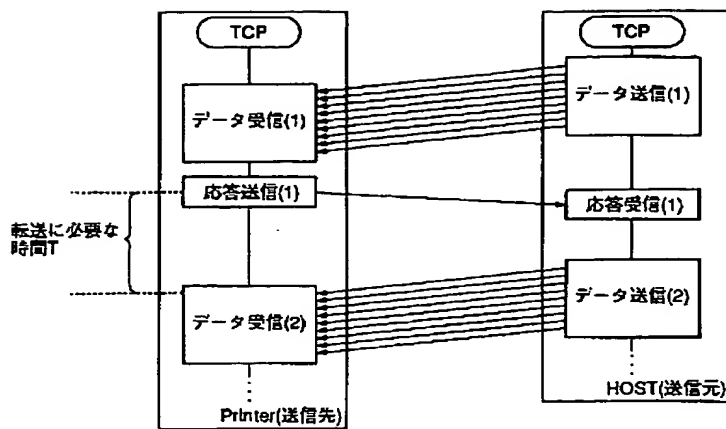
プリンタ言語判別部とジョブ制御モジュールとの間のデータ受け渡し処理の説明図

【図 25】



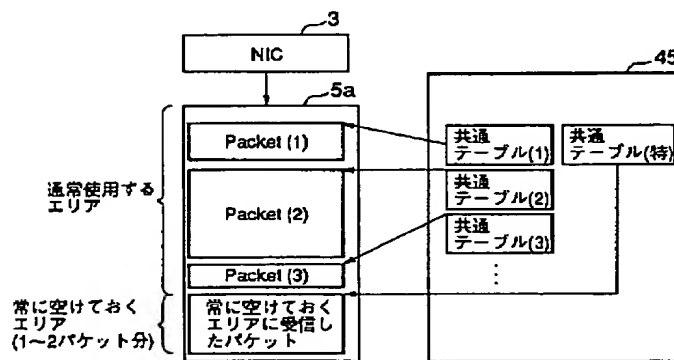
第8の実施の形態を示すブロック図

【図24】



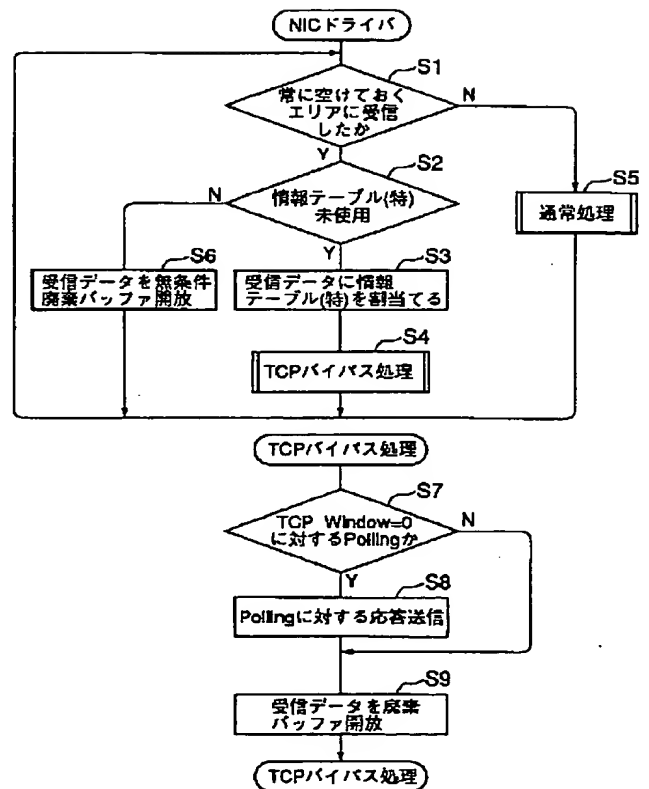
TCP間のデータ受け渡し処理の説明図

【図26】



第9の実施の形態を示すブロック図

【図27】



第9の実施の形態の動作を示すフローチャート

**This Page is Inserted by IFW Indexing and Scanning
Operations and is not part of the Official Record**

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☐ FADED TEXT OR DRAWING
- ☐ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☐ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☒ OTHER: SMALL Text

IMAGES ARE BEST AVAILABLE COPY.

As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.